Computing Length-Preserved Free Boundary for Quasi-Developable Mesh Segmentation

Charlie C.L. Wang, *Member*, *IEEE*

Abstract—Stretch-free surface flattening has been requested by a variety of applications. At present, the most difficult problem is how to segment a given model into nearly developable atlases so that a nearly stretch-free flattening can be computed. The criterion for segmentation is needed to evaluate the possibility of flattening a given surface patch, which should be fast computed. In this paper, we present a method to compute the length-preserved free boundary (LPFB) of a mesh patch which speeds up the mesh parameterization. The distortion on parameterization can then be employed as the criterion in a trial-and-error algorithm for segmenting a given model into nearly developable atlases. The computation of LPFB is formulated as a numerical optimization problem in the angle space, where we are trying to optimize the angle excesses on the boundary while preserving the constraints derived from the *closed-path theorem* and the length preservation.

Index Terms— Boundary representations; Geometric algorithms, languages, and systems.

1 Introduction

THE request of computing a stretch-free parameterization, via surface flattening, has been arisen by numerous applications in computer graphics (e.g., texture mapping [17], [27], [28], [29], [44], [45], [47], [48], [49], medical imaging [30], [36], toy fabrication [13], [21], [32], and computer-aided design [1], [2], [3], [22], [39], [41]), where a corresponding 2D region D for a given surface patch P is determined. At present, the focus of interests has moved to how to segment a given model M into surface patches each of which can be flattened with less distortion. An ideal surface flattening (parameterization) preserves the distances between any two points on P and D – mathematically named as *isometric mapping*, and also preserves the areas on P and D. From differential geometry [8], we know that only fully developable surface, which is a small class among all freeform surfaces, can be flattened to give an isometric mapping. Therefore, to be practical, a method is requested to segment M into nearly developable (i.e., quasi-developable) patches instead of fully developable ones. One application for the patches like this is the toy fabrication (as shown in Fig.1 of [13] and Fig.10 in this paper), where small number of patches are desired since more patches yield more stitching work in the fabrication. Most recent attempts for this are [13] and [32], where the authors partitioned the given model into several quasi-conical patches. However, conical surface is not the only type of a developable surface. A more general criterion for surface flattening is needed. This is the motivation of our work – computing length-preserved free boundary (LPFB) to speed up the parameterization so that the distortions on the resultant planar mesh can serve as the criteria for quasi-developable mesh segmentation.

The most straightforward criterion for developability is the integral, $\sigma = \int |K| dA$, of Gaussian curvature K or any discrete form of its variants [23], [33] and [43] over the given surface patch P. However, although discrete Gaussian curvature works well on dart insertion to reduce the stretch of flattening where the local developability is counted, it is weak at distinguishing the degree of global developability on nearly developable surfaces. For the examples shown in Fig. 1, both are with very few non-developable vertices giving *non-zero* Gaussian curvature, and the value of Gaussian curvature integral on the cylinder is greater than the cube. However, the stretch on the cube is even more significant, which can be detected from the checkboard texture. Furthermore, using σ as criterion cannot prevent generating the non-flattenable patches led by topological obstructions (e.g., a cylinder cannot be flattenable without inserting a cut to link its two openings). Note that as working on models with polygonal meshes, the definition of *developability* is generalized – simply, the less stretch is given on a flattened mesh surface, the higher developability is with the surface. On the other hand, even if the fastest linear parameterization approach (e.g., [19] and [7]) is conducted, computing the flattening with free boundaries is still time-consuming. Therefore, it is impractical to directly employ it in a trial-and-error algorithm to segment a model into nearly developable patches, where the measurement of developability needs to be iteratively computed for numerous times. We would like to request a criterion for developable mesh segmentation which holds the following properties:

- Effectiveness – the criterion should consider the developability globally, so that the nearly developable surfaces (e.g., the cylinder in Fig. 1) with few non-developable vertices can be distinguished from those non-developable patches (e.g., the cube in Fig. 1);
- Efficiency – as will be repeatedly evaluated when separating the model into nearly developable atlases, the criterion should be computed efficiently.

From observation, we find that for a fully developable sur-

The author is with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong, P. R. China
E-mail: cwang@mae.cuhk.edu.hk; Fax: (852)2603 6002; Tel: (852)2609 8052

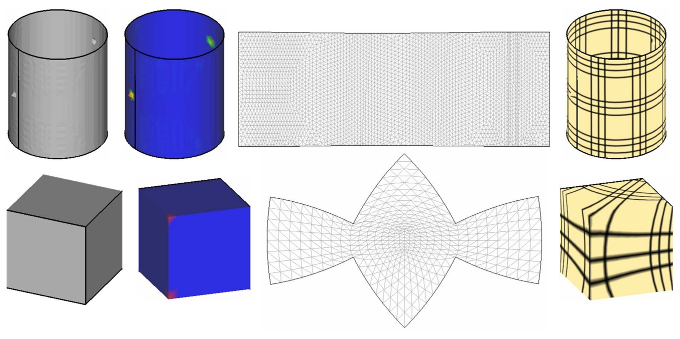


Fig. 1. Two examples both have few vertices with non-zero Gaussian curvature; however, one is nearly developable (the noisy cylinder – the sharp points are noises) while another (the cube with the given five cuts) is non-developable, where the black curves are cutting paths. The second column images show the Gaussian curvature map, where blue color represents zero Gaussian curvature and red denotes the maximal value – the two corner vertices on the back of the cube are with high Gaussian curvature since no cut passes them. The third column gives the length-preserved free boundaries computed by our approach and their corresponding surface flattening. The last column illustrates the texture checkboard determined from the flattening – the smaller distortion is shown, the higher developability is with the surface. The integral of discrete Gaussian curvature (which is computed by [23]) over area on the cylinder example is 8.930, which is even greater than the integral result 3.142 on the cube example – but the cylinder can obviously be flattened with less stretch.

face P, both the area and the edge lengths on its flattening D are coherent with P. Denoting the boundaries of P and D as ∂P and ∂D, $\|\partial P\| = \|\partial D\|$ is preserved on fully developable surfaces. For a general surface P, by computing a optimal length-preserved free boundary ∂D (in short, LPFB) which minimizes the angle distortion between ∂P and ∂D while preserving $\|\partial P\| = \|\partial D\|$, the distortion on the flattening of P with ∂D fixed can serve as a good indicator for the developability – the less distortion, the more developable patch P is. Experimental tests show that fixing LPFB acts as an amplifier for detecting the developability of a surface by the distortion on D. More specifically, for quasi-developable surface (like the cylinder in Fig.1), fixing LPFB reduces the unexpected distortion from noisy points; on the other hand, it enlarges the distortion on non-developable surfaces (as the cube in Fig.1). Moreover, fixing ∂D speeds up the computation D since we can decouple the x- and y- coordinates to reduce the dimension of linear system.

The approach presented in this paper computes the length-preserved free boundary (LPFB) and employs it to accelerate the intrinsic parameterization (IP) of interior mesh regions, where the distortions on resultant 2D meshes are used in a trial-and-error approach as criteria for segmenting a given 3D mesh into quasi-developable atlases.

1.1 Previous Work

The work proposed in this paper relates to several previous researches in the areas of: surface parameterization, texture atlas generation, and surface flattening for pattern design, which are consecutively reviewed below.

As having a lot of applications in various fields of science and engineering, surface parameterizations have been studied for many years. The parameterization of a given three-dimensional surface P computes its corresponding 2D parametric domain D, usually via surface flattening. An ideal surface flattening preserves the distances between any two points on P and D – mathematically named as *isometric mapping*. Another very important property related to the surface flattening is that the bijectivity always needs to be satisfied on the mapping Ω as $\Omega : P \to D$ and $\Omega^{-1} : D \to P$. However for practical applications, neither the isometric nor the bijectivity can be always satisfied. Therefore, a surface parameterization always introduces distortion in either angles or areas. All parameterization approaches in literature give strength on how to minimize the distortions in some sense, which has been mentioned in the detail review [10] by Floater and Hormann.

Only a few parameterization schemes can generate a planar domain with a free boundary (e.g., [7], [12], [14], [18], [19], [34], [35] and [46]). Among them, only [14], [34] and [35] are the free-boundary methods that guarantee non-local-overlap on the parameterization. The Angle Based Flattening (ABF) method presented in [34] defined an angle preservation metric, by which the parameterization was computed in the angle space and then converted into 2D coordinate. When converting the parameterization from the angle space into planar coordinates, natural boundaries were automatically formed. Being nonlinear, the basic ABF method is relatively slow. The basic ABF method was extended into the ABF++ approach in [35], where the speed is improved by some numerical techniques and the hierarchical flattening technique. Borrowing the idea from [35], we compute the length-preserved free boundary in the angle space, which is also angle-based non-linear optimization. However, the dimension of our non-linear system is much smaller than [34] and [35] so that can be computed faster. Being linear, the approach of Lévy [19] and Desbrun et al. [7] are the fastest parameterizations which also give free boundaries. It is difficult to add special constraints about the shearing (for shape) or stretching (for length) on boundaries in their linear systems. However, the positions of boundary vertices are more important than interior vertices since the shape and the area of a planar domain are essentially defined by the boundary vertices. In [18], the authors conducted a virtual boundary technique to generate a parameterization with free boundaries. Recently, this problem has been further studied in [14] to ensure the free boundaries yielding a planar embedding (i.e., no flipped triangle). The approach of Zayer et al. in [46] consecutively compute a conformal map with a fixed boundary, a boundary-free conformal map, and a boundary-free quasi-harmonic map to improve the quality of a parameterization. Three linear equation systems need to be solved in their approach, so it will not be as fast as ours (with only one linear equation system). [11] and [12] are also related to our work. The method of [12] is based on measuring the conformality of a (non-degenerate) bivariate linear function by the condition number of its Jacobian with respect to the Frobenius-norm. The computation is based on a non-linear minimization, therefore is relative slow. Gu and Yau in [11] compute global conformal parameterization on surfaces with nontrivial topologies. Their output parameterization preserves the conformality everywhere except for a few points. None of above parameterization approaches addresses the problem of computing *length-preserved* free boundary.

In order to generate a texture mapping with less distor-

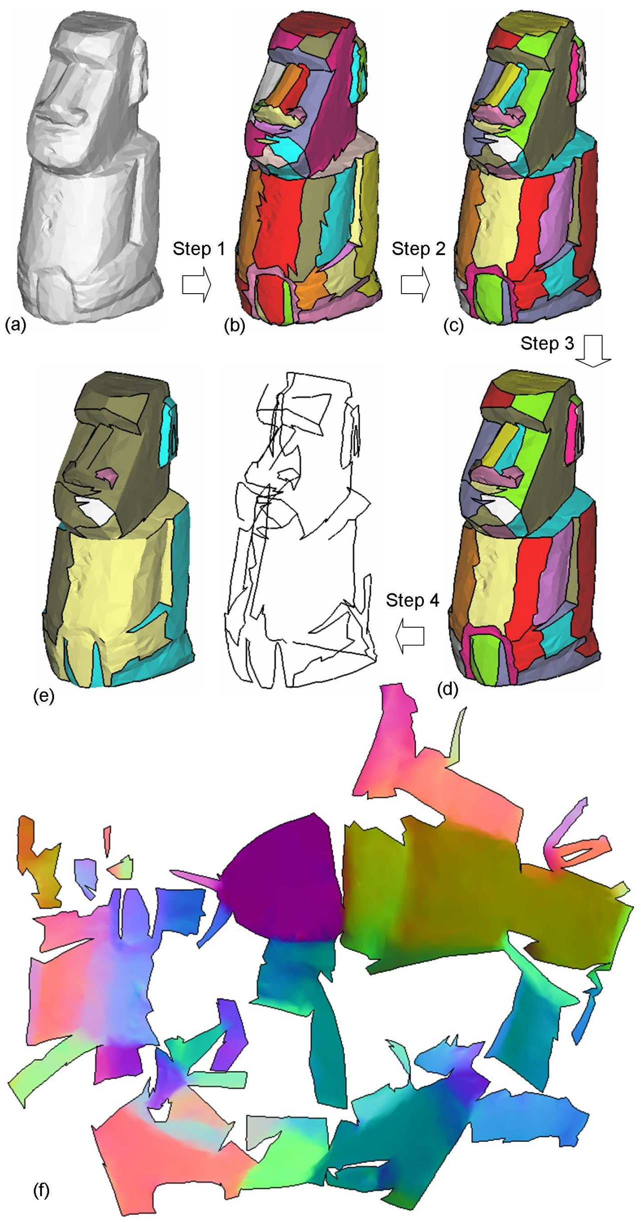


Fig. 2. Illustration for the quasi-developable mesh segmentation algorithm: (a) the given model, (b) after error controlled VSA (with 68 patches), (c) after minimum-cut based boundary refinement (still with 68 patches), (d) after boundary denoising, (e) the result from LPFB based patch merging (with 7 patches), and (f) the surface flattening results where colors represent normal vectors on the original 3D model.

tion, many methods have employed mesh segmentations to generate texture atlases. Some of the segmentation methods, such as [15], [16] and [47], focus on feature-based segmentation. They segmented the given model into meaningful components, which are typically far from being developable. In [19], Lévy et al. segmented the model into texture atlases using crease lines as boundaries. Charts are then further segmented if the stretch after the parameterization is too high. The authors in [37] presented a similar idea but in a triangle-based on-the-fly approach – it is slow. The approach presented in [48] segments a mesh model in a similar manner but with a different surface parameterization method [49] based on multidimensional scaling (MDS). These are all trial-and-error methods, where a more efficient method for measuring developability is desired to speed up the partition. Some of the segmentation methods constructed nearly planar charts [4], [20], [28], [29], and [42]. Although planar charts are developable, most developable surfaces are obviously not planar. Therefore, planar segmentations usually result in more charts than necessary. Julius et al. [13] and Shatz et al. [32] developed algorithms to separate a given model into quasi-conical proxies. Again, it is a sufficient (but not necessary) condition that a conical surface is developable. A more general criterion for developable surface is needed. Recently, in [43], the authors introduced a mesh segmentation algorithm based on the integral of Gaussian areas over a patch. However, this measurement cannot distinguish the developability on the patches shown in Fig. 1.

In the area of computer-aided design, the surface flattening for pattern design has been studied in various industries (see [1], [2], [3], [22], [39], [40], and [41]). All these approaches have the common drawback – the computation is very time-consuming. Therefore, it is not appropriate for using them to repeatedly evaluate the developability.

1.2 Contribution

We conduct a boundary mapping technique to compute the length-preserved free boundary (LPFB) which minimizes angle distortion while preserving the length of boundary edges. The LPFB is employed to accelerate the classical parameterization for the interior mesh region using cotangent weights, where the distortions on resultant 2D meshes are conducted in a new trial-and-error algorithm to segment a 3D mesh into quasi-developable atlases.

The computation of LPFB together with the followed parameterization (by decoupling two planar coordinates) is even faster than the fastest flattening approach in literature (e.g., [19] and [7]) – thus the requirement of efficiency is satisfied.

Our method for computing LPFB overcomes the limitation of disk-like topology – LPFB of a surface with multiple loops can be successfully computed.

2 Quasi-Developable Mesh Segmentation

This section gives the overview of our trial-and-error algorithm for segmenting a given mesh model H into nearly developable patches. The basic idea is that: we firstly segment H into nearly planar charts; then the nearly planar charts are incrementally merged back into larger quasi-developable surface patches through a trial-and-error procedure where the merging criteria are efficiently evaluated with the help of LPFB. As illustrated in Fig. 2, the segmentation algorithm consists of four steps.

Step 1: Error controlled Variational Shape Approximation

First of all, a given model H is segmented into nearly planar patches by the Variational Shape Approximation (VSA) algorithm [4] – here we control the shape approximation error instead of the proxy number. Starting from one seed, we incrementally add more seeds into the k-proxy clustering algorithm until the maximal approximation error $\max L^{2,1}$ shown on all charts is less than a given tolerance. The error controlled VSA results in a number of small patches on a complex model.

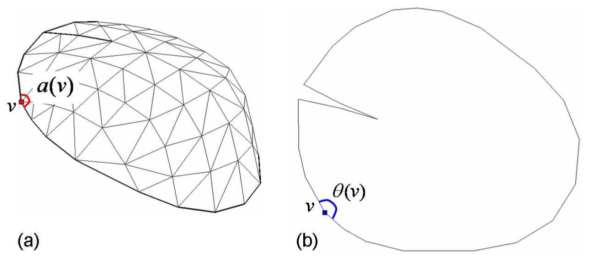


Fig. 3. Illustration of the surface inner angle and the inner turning angle at a boundary vertex: (a) the given triangular mesh patch P and (b) the boundary of its corresponding planar domain D.

Step 2: Minimum-Cut based boundary refinement

In this step, the boundaries of patches are refined by the minimum-cut method akin to [16] so that the zigzag effects are improved. Briefly, a fuzzy area Γ is determined around the boundaries between different charts. After converting the faces in Γ to the nodes of a weighted graph, the repartition of triangles is found by a minimal cut algorithm on the graph weighted by dihedral angles (ref. [5]).

Step 3 (optional): LSE based boundary denoising

In order to reduce the effect from noises, we further adjust the vertices located on the refined boundaries. This is an optional step – those noise-free models do not need this. For the faces f on a chart around a boundary vertex, the normal vectors n_f are assigned as the chart normal vector, which is the normalized vector by summing area weighted triangles normals on the chart (see [4] for details). Then, the position of every boundary vertex v_i is updated by minimizing the following least square error (LSE) defined on the faces

$$E(v_i)=\sum_{j\in star(i)}\sum_{f\in F_{ij}}(n_f\cdot(v_i-v_j))^2$$

where $star(i)$ denotes the 1-ring neighboring vertices of v_i and F_{ij} represents the two faces that are adjacent to the edge v_iv_j. $E(v_i)$ can be iteratively minimized by the following update given in [38]

$$v_i'\leftarrow v_i+\lambda\sum_{j\in star(i)}\sum_{f\in F_{ij}}n_f n_f^T(v_j-v_i)\,.$$

Step 4: LPFB based patch merging

This step shows the major contribution of the proposed approach. The neighboring small charts are incrementally merged into a larger patch if the new patch P is still quasi-developable. We firstly compute the LPFB of P (the computation method will be detailed in section 3) and then employ the intrinsic parameterization [7] to determine the flattening result D while fixing the boundary vertices. The distortions on D are employed to evaluate the developability. The detail LPFB based patch merging will be presented in section 4.

3 LPFB Computing

3.1 Basic Method

The basic methodology for computing a length-preserved free boundary (LPFB) is presented in this section. The problem for determining the planar coordinates of boundary vertices on ∂D is going to be formulated as a constrained optimization problem, where the objective functional is derived from the following two energy terms for the differences between ∂D and ∂P.

Boundary Length Energy – From the length-preservation requirement between ∂D and ∂P, we define the first energy term by the edge length on boundaries. For a polygonal edge e on ∂P, letting l_e^0 and l_e representing its corresponding length on ∂P and ∂D, the boundary length energy is defined as

$$\Pi_L=\frac{1}{n}\sum_{e\in\partial P}\left|l_e-l_e^0\right| \qquad (1)$$

where n presents the number of edges on ∂P. The boundary length energy is straightforward. However, since $\partial D\in\mathbf{R}^2$ and $\partial P\in\mathbf{R}^3$, it is not easy to measure the shape similarity between ∂D and ∂P by Π_L. Therefore, the second energy term is introduced.

Boundary Morphological Energy – We employ angles to measure the morphological difference between ∂D and ∂P. For a vertex v on ∂P, suppose that its surface inner angle on P is a_v and its inner turning angle on D is θ_v (see the illustration in Fig. 3), the shape similarity between ∂D and ∂P is evaluated with the help of the following boundary angle error together with the boundary length error.

$$\Pi_\theta=\frac{1}{n}\sum_{v\in\partial P}\left|a_v-\theta_v\right| \qquad (2)$$

In Π_θ, n is the number of vertices on ∂P which is actually same as the number of edge on ∂P, and the surface inner angle α_v is the sum of all vertex angles at v on P.

Based on these boundary energy terms, we can compute an optimal LPFB of P by the functional

$$\arg\min_{v\in\partial D}\Pi_\theta \quad s.t.\,\Pi_L=0 \qquad (3)$$

However, as both Π_L and Π_θ are complex in term of the planar position of boundary vertices, directly solving the problem defined in Eq.(3) is quite slow thus not practical. The optimization functional needs to be reformulated. By observation, we find that solving the above optimization problem in the angle space could greatly simplify the formulas. Then, the morphological term is converted into

$$J_E=\sum_i\frac{1}{2}(\theta_i-a_i)^2 \qquad (4)$$

where θ_i is the inner turning angle of a boundary vertex $v_i\in\partial P$, a_i is the surface inner angle v_i on P, and the index i of boundary vertices is given anti-clockwise.

From the *closed-path theorem* (ref. [24]), we know that: for a simple non-self-intersection planar closed path, if its path is anti-clockwise, the total turning is 2π. As shown in Fig. 4, its total turning by accumulating vertex turning angles is

$$\sum_{i=1}^n(\pi-\theta_i)\,,$$

so the constraint below must be satisfied in the angle space

$$n\pi-\sum_{i=1}^n\theta_i\equiv 2\pi \qquad (5)$$

Besides J_E and the constraint comes from the closed-path theorem, we still need to add position coincident constraints on some boundary vertices. After determining the

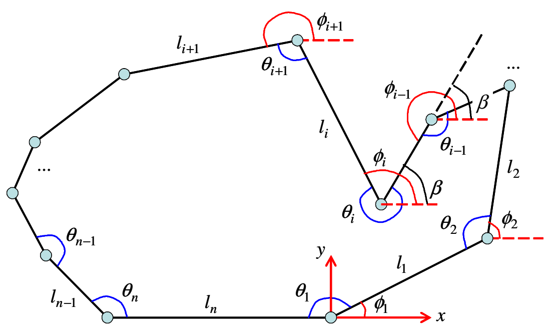


Fig. 4. Closed-path constraints on the planar boundary.

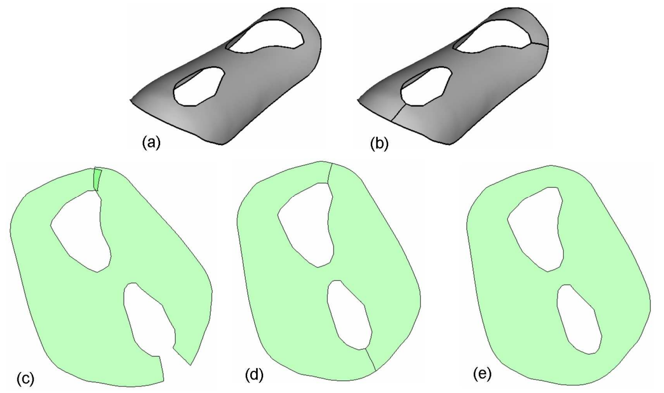


Fig.5. Virtual-cutting scheme for surface patch with multiple boundary loops: (a) the given mesh patch P, (b) a duplicated patch P_d of P is generated and cut by the shortest path from inner loops to the outer boundary, (c) LPFB of P_d determined by applying the basic method, (d) LPFB obtained by adding position constraints of the vertices on the cutting path, and (e) the resultant D computed by the virtual cutting scheme.

inner turning angles θ_i s and placing v_1 at the origin, the planar coordinate (x_i, y_i) of a boundary vertex v_i becomes

$$x_i = \sum_{k=1}^{i-1} l_k \cos\phi_k , \quad y_i = \sum_{k=1}^{i-1} l_k \sin\phi_k . \tag{6}$$

As been illustrated in Fig. 4, we have $\theta_i = 2\pi - (\phi_i - \beta)$ at the vertex v_i with $\beta = \phi_{i-1} - \pi$, which leads to $\phi_i = \pi - \theta_i + \phi_{i-1}$. Together with $\phi_1 = \pi - \theta_1$, the general formula for ϕ_i can be derived in terms of θ_i as

$$\phi_i = i\pi - \sum_{b=1}^{i} \theta_b . \tag{7}$$

In order to ensure ∂D be closed, we must let (x_{n+1}, y_{n+1}) be coincident with the origin, which leads to

$$\sum_{i=1}^{n} l_i \cos\phi_i \equiv 0, \quad \sum_{i=1}^{n} l_i \sin\phi_i \equiv 0 . \tag{8}$$

Therefore, LPFB of a given patch P can be determined by the following constrained optimization problem defined in the angle space.

$$\begin{gathered} \arg\min_{\theta_i} \left\{ \sum_i \frac{1}{2} (\theta_i - a_i)^2 \right\} \\ s.t. \;\; n\pi - \sum_{i=1}^{n} \theta_i \equiv 2\pi, \sum_{i=1}^{n} l_i \cos\phi_i \equiv 0 , \sum_{i=1}^{n} l_i \sin\phi_i \equiv 0 \end{gathered} \tag{9}$$

The efficient numerical implementation for solving Eq.(9) will be given in section 3.3. After obtaining the optimal θ_i s, the planar coordinates of boundary vertices can be easily computed by Eq.(6) and (7). Eq.(6) and (8) here are similar to the 2D blending approach presented in [31], but [31] neither explicitly constrained the self-intersection nor preserved edge lengths as what we give here.

3.2 Virtual-Cutting Scheme

The basic method above works well for a mesh surface with disk-like topology (e.g., the one in Fig. 3); however, it shows problems when computing LPFB of a surface with multiple boundary loops. Therefore, the virtual cutting-scheme is developed below to solve this problem. The algorithm consists of six steps (illustrated with the example in Fig. 5):

1. Constructing a duplicated patch P_d of P, where each vertex $v_i' \in P_d$ has a corresponding vertex $v_i \in P$;
2. For every inner boundary loop on P_d , a shortest path from it to the outer loop is determined by the Dijkstra's algorithm [5] with multiple sources;
3. Cutting P_d along the shortest paths (see Fig.5(b));
4. Applying the basic method for LPFB on P_d (resulting in Fig.5(c));
5. Adding the coincident constraints of the vertices on cutting paths into the optimization framework – so that an improved LPFB is obtained (as Fig.5(d));
6. By the correspondences between $v_i' \in P_d$ and $v_i \in P$, the planar coordinate for all boundary vertices on P can be determined (see Fig.5(e)).

In the following, we will describe the technical details of this scheme.

Step 1 and 2 are trivial. In step 3, we iteratively introduce duplicated edges on the edges belonging to the cutting path, while the coincident pairs are stored as the information will be used in step 5. Step 4 takes the basic method for computing LPFB – for the example shown in Fig. 5, the example result after this step is as Fig. 5(c), where the newly created cutting vertices are not coincident with their parents. After getting result from the basic method, in step 5, we will further deform the boundary to make the vertices stored in coincident pairs become coincident. For this purpose, the constraints defined in Eq.(9) needs to be adjusted. Suppose $\alpha(p)$ and $\beta(p)$ represent the indices of the vertices in the pth coincident pair, without loss of generality, letting $\alpha(p) < \beta(p)$, the coincident constraint of $v_{\alpha(p)}$ and $v_{\beta(p)}$ yields

$$\sum_{k=\alpha(p)}^{\beta(p)-1} l_k \cos\phi_k \equiv 0, \quad \sum_{k=\alpha(p)}^{\beta(p)-1} l_k \sin\phi_k \equiv 0 ,$$

by Eq.(6). Together with $\alpha(0) = 1$ and $\beta(0) = n+1$, the constrained optimization problem in Eq.(9) is reformulated to

$$\begin{gathered} \arg\min_{\theta_i} \left\{ \sum_i \frac{1}{2} (\theta_i - a_i)^2 \right\} \\ s.t. \;\; n\pi - \sum_{i=1}^{n} \theta_i \equiv 2\pi, \sum_{k=\alpha(p)}^{\beta(p)-1} l_k \cos\phi_k \equiv 0, \sum_{k=\alpha(p)}^{\beta(p)-1} l_k \sin\phi_k \equiv 0, \cdots \end{gathered} \tag{10}$$

where $0 \le p \le m$ with m pairs of coincident vertices that were constructed in step 3.

After solving the optimization problem in Eq.(10), a LPFB is determined with all the child-and-parent vertices on cutting paths coincident (e.g., see Fig. 5(d)). The planar coordinates of vertices on ∂P are then computed in the last step from the corresponding vertices on ∂P_d.

3.3 Efficient Numerical Implementation

The only left problem for computing LPFB on a given mesh patch is how to efficiently solve the constrained optimization problem defined above. Using the Lagrange multiplier

$$(\lambda_\theta, \lambda_{0x}, \lambda_{0y}, \ldots, \lambda_{px}, \lambda_{py}, \ldots, \lambda_{mx}, \lambda_{my}),$$

the constrained optimization problem can be converted into an augmented objective function

$$J(X) = J(\theta_1, \ldots, \theta_n, \lambda_\theta, \lambda_{0x}, \lambda_{0y}, \ldots, \lambda_{px}, \lambda_{py} \ldots, \lambda_{mx}, \lambda_{my});$$

in detail,

$$\begin{aligned} J(X) = \sum_{i=1}^{n} \frac{1}{2}(\theta_i - a_i)^2 + \lambda_\theta((n-2)\pi - \sum_{k=1}^{n} \theta_k) \\ + \sum_{p=0}^{m} \lambda_{px} \sum_{k=\alpha(p)}^{\beta(p)-1} l_k \cos\phi_k + \sum_{p=0}^{m} \lambda_{py} \sum_{k=\alpha(p)}^{\beta(p)-1} l_k \sin\phi_k \end{aligned}, \quad (11)$$

which can be minimized by the Newton's method [25].

```
while $\|\nabla J(X)\| > 10^{-5}$
    solve $\nabla^2 J(X)\delta = -\nabla J(X)$;
    $X \leftarrow X + \delta$;
end
```

The size of Hessian matrix $\nabla^2 J(X)$ is n+2m+3. For a complex model, n in general is a large number so that it is time-consuming to solve the linear equation system. Borrowing the idea for speeding up a nonlinear optimization in [35], we conduct the sequential linearly constrained programming [25] to minimize $J(X)$ by neglecting the terms coming from the second derivatives of the constraints in the Hessian matrix $\nabla^2 J(X)$. The equation

$$\nabla^2 J(X)\delta = -\nabla J(X)$$

solved at each step is simplified into

$$\begin{bmatrix} \mathrm{I} & \Lambda^T \\ \Lambda & 0 \end{bmatrix} \begin{bmatrix} \delta_\theta \\ \delta_\lambda \end{bmatrix} = \begin{bmatrix} B_\theta \\ B_\lambda \end{bmatrix} \quad (12)$$

with $X = (\theta_1, \ldots, \theta_n, \lambda_\theta, \lambda_{0x}, \lambda_{0y}, \ldots, \lambda_{px}, \lambda_{py} \ldots, \lambda_{mx}, \lambda_{my})$. In this equation, Λ, B_θ and B_λ can all be efficiently evaluated (see Appendix). Eq.(12) can then be solved by

$$\Lambda\Lambda^T \delta_\lambda = \Lambda B_\theta - B_\lambda \quad (13)$$

$$\delta_\theta = B_\theta - \Lambda^T \delta_\lambda \quad (14)$$

where $\Lambda\Lambda^T$ is $(2m+3)\times(2m+3)$ as Λ is $(2m+3)\times n$. Since m is usually small ($m << n$), Eq.(13) can be efficiently solved using *Gaussian elimination* if $rank(\Lambda\Lambda^T) = 2m+3$, or it can be solved by *singular value decomposition* (SVD) if not full rank. By δ_λ determined in Eq.(13), δ_θ can be easily computed from Eq.(14) – thus $\delta = (\delta_\theta^T, \delta_\lambda^T)$ is determined.

The same as all other nonlinear problems, giving a good initial value can speed up the above computation. In our implement, we choose $\theta_i = a_i$ and $\lambda_\theta = \lambda_{px} = \lambda_{py} = 1$. The computation usually converges in steps of tens.

4 LPFB-Based Patch Merging

In this section, after introducing the measurements for classifying patches, details of the LPFB-based patch merging algorithm are presented.

4.1 Measurements for Classification

The measurements employed in our segmentation algorithm all rely on the flattening Ω of P in D with a fixed LPFB. For this purpose, we conduct the following linear equation system from the *intrinsic parameterization* method [7] to calculate the planar coordinates on interior vertices.

$$\mathrm{M}U = V \quad (15)$$

where U is the vector of planar coordinate, V is the vector for boundary condition that

$$V_i = \begin{cases} \psi(i), & i \in \partial P \\ 0, & i \notin \partial P \end{cases} \quad (16)$$

with $\psi(i)$ representing the planar coordinate of vertex v_i on the LPFB of P, and M is a sparse matrix whose coefficients are given by

$$\mathrm{M}_{ij,i\notin\partial P} = \begin{cases} \cot(\gamma_{ij}) + \cot(\xi_{ij}), & j \in star(i) \\ -\sum_{k\in star(i)} \mathrm{M}_{ik}, & i = j \\ 0, & otherwise \end{cases}. \quad (17)$$

$$\mathrm{M}_{ij,i\in\partial P} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}. \quad (18)$$

In Eq.(17), $star(i)$ denotes the 1-ring neighbors of vertex v_i, and γ_{ij} and ξ_{ij} represent the opposite angles in the two neighboring triangles of the edge v_iv_j on P respectively. Why this computation is faster than directly applying the intrinsic parameterization? The reason is that for a surface P with n vertices the dimension of the linear system in Eq.(15) is $n \times n$ since we can separately compute the x- and y-coordinates. However, for the method in [7], when free boundary is needed, both x- and y-coordinate should be computed together in a $2n \times 2n$ linear system. The lower-bound of the computing time for solving sparse linear system is linear – therefore half of the computing time is reduced; however, in general, it depends on the pattern of the matrix M. With the conjugate gradient solver, the upper bound is in quadratic complexity. Our tests shows that by reducing the dimension of M in half, the speed up is usually much more than half. Detail statistics will be shown in section 5.

By Ω determined from LPFB of P, a hybrid classification using three measurements is conducted. The measurements include: the area distortion $A(\Omega)$, the $L^2(\Omega)$ norm of texture stretch, and the global self-overlapping $S(\Omega)$. The area distortion computes the percentage of area change in D and P by

$$A(\Omega) = \frac{\left|\sum_T A_T - \sum_T A_T^0\right|}{\sum_T A_T^0} \times 100\% \quad (19)$$

where A_T and A_T^0 are respectively the areas of a triangle $T \in P$ in 2D and 3D. Although with a small $A(\Omega)$ is the necessary condition for P being quasi-developable, it is possible to have extreme cases that small $A(\Omega)$ is given on a non-developable surface. Therefore, the L^2 norm of texture stretches from [28] is conducted to exclude these extreme cases. It measures the surface distortion on Ω to P, where 1.0 is the lower bound on any parameterization. Above two measurements cannot distinguish the global self-overlapping on Ω (e.g., the self-overlapping in Fig. 5(c)).

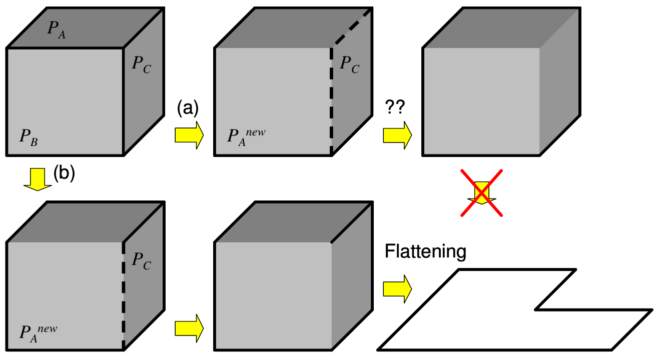


Fig. 6. Comparison for the merging strategy: (a) patch-based vs. (b) boundary-based.

Numerically detecting the self-overlapping is usually time-consuming. To speed-up, here we seek help from the graphics hardware. After drawing all triangles in the blending mode with the value of transparency 0.25, the pixel drawn more than once (so that leads to overlap) can be easily detected by its pixel color value. When overlapping is found, the measurement $S(\Omega)$ gives 1; otherwise, $S(\Omega)=0$ is returned. As long as the resolution of discretization is high enough, the self-intersection can be effectively detected in a given tolerance.

4.2 Merging Algorithm

We incrementally merge neighboring small patches into a larger patch if the new patch can still be flattened without stretching. Therefore, the measurements for classification need to be repeatedly evaluated – our LPFB-based technique shows its strength on the speed here.

The merging can be conducted in the patch-based or the boundary-based manner. We find that the boundary-based method can generate patches with lager areas comparing to the patch-based merging when adopting the same criteria. The reason can be simply explained by the example shown in Fig. 6. For the patch-based approach, after merging patch P_A and P_B into a new P_A^{new}, the two patches P_A^{new} and P_C cannot pass the developability test since their merging will lead to a non-developable surface. However, if the boundary-based strategy is conducted, after eliminating the boundary between P_A and P_B, we can still merge P_C into P_A^{new} by removing the boundary curve between P_B and P_C (see Fig. 6). Note that the dangling edges are allowed in our approach, which will be converted into cuts when generating charts from the segmentation results.

The following boundary-based merging algorithm is conducted to merge small nearly planar patches into larger nearly developable patches.

1. Every boundary curve between different patches is inserted into a maximum heap Ψ which sorts the area of two patches aside the boundary curve;
2. If Ψ is empty, stop the algorithm;
3. Remove the top curve ζ from Ψ;
4. Duplicate a patch P simulating the merged patch by eliminating ζ;
5. Compute the LPFB and then the parameterization Ω of P from the fixed LPFB;
6. Evaluate $A(\Omega)$, if $A(\Omega)>10\%$ go back to step 2;
7. Evaluate $L^2(\Omega)$, if $L^2(\Omega)>2.0$ go back to step 2;
8. Evaluate $S(\Omega)$, if $S(\Omega)=1$ go back to step 2;
9. Merge the patches aside ζ into a new one P^{new} by removing ζ;
10. For the remained boundary curves on P^{new}, update their positions in the heap Ψ;
11. Go back to step 2.

This algorithm can efficiently merge small planar patches into nearly developable large patches. The two thresholds, 10% and 2.0, are determined from a supervised learning process [9] with a set of training mesh surfaces. Figure 2(d)-(e) show an example for the LPFB-based patch merging, and more examples will be given in the following section.

5 Results and Discussion

The computing method for length-preserved free boundary has been tested on several freeform mesh surfaces, and compared with the intrinsic parameterization [7] (IP), the least squares conformal maps [19] (LSCM), and the angle-based flattening [34] (ABF) in several aspects – including the computing time, the area change, the texture distortion, and the variation of boundary length and shape on the flattening results (see Table 1). The first two examples tested are the cylinder and the cube which have already been shown in Fig.1. The cylinder model is a quasi-developable surface (i.e., a developable surface with few non-developable noisy vertices), and the cube is absolutely a non-developable surface. The results from our method shown in Fig.1 can be successfully distinguished by the area variation (see the statistics in Table 1). However, results of both the cylinder and the cube from IP (and LSCM) are greatly distorted (see Fig.7). This is because that they minimize the angle variation on all vertices, so that noisy vertices on the cylinder affect the computation to yield great distortion. It is hard to distinguish them by the results from IP and LSCM, which has also been proved by the statistics shown in Table 1. Note that we fix the two endpoints of the longest boundary edge when solving the linear systems in IP and LSCM. Our results are quite similar to the results of ABF; however the computation time of ABF is much longer. In the third example – the two-hole patch (see Fig. 5 and 8), all four approaches give the similar results but ours generates the most similar result to ABF (see the statistics in Table 1). The following example is a mesh with about 27k triangles which is generated by the segmentation algorithm presented in this paper. Again, our result is similar to ABF but much faster.

From the statistics listed in Table 1, it is easy to find that our method generate results very close to ABF, IP and LSCM on quasi-developable meshes. While on the non-developable mesh (e.g., the cube in Fig. 1 and 7), as mentioned at the beginning of this paper, fixing LPFB acts as an amplifier which enlarges the distortion. This characteristic satisfies the effectiveness property requested for the developable mesh segmentation criterion. Now, let us take a look at the computational time. The time needed in our approach is much less than the other three methods. We solve the sparse linear systems in IP and LPFB+IP by the *Preconditioned Biconjugate Gradient* method from [26]. The code for LSCM is downloaded from the website: http://www.loria.fr/~levy/, where the sparse linear system is solved by the *Jacobi Preconditioned Conjugate Gradient* method. The results for ABF are generated by the software

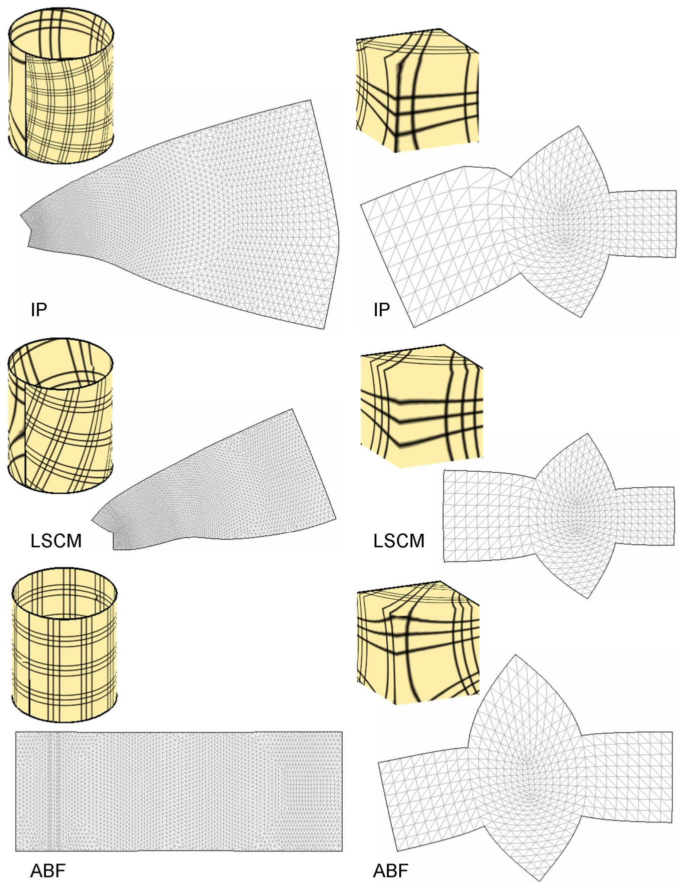


Fig. 7. The flattening results (and their corresponding checkboard display) of the two models previously shown in Fig. 1 are determined by IP [7], LSCM [19], and ABF [34].

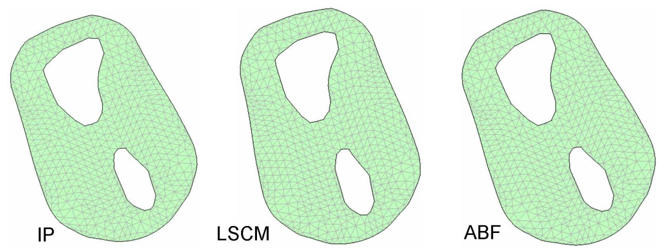


Fig. 8. The results of IP, LSCM and ABF on the two-hole patch that has been previously shown in Fig. 5.

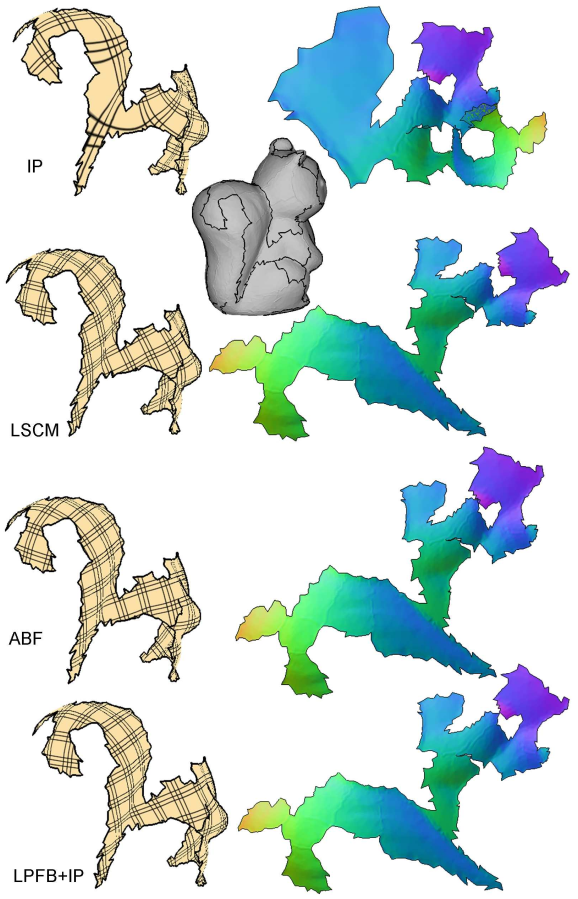


Fig. 9. The flattening results comparison on a large mesh with more than 27K faces. The 3D mesh patch is generated by our segmentation algorithm from the squirrel model.

Graphite which is also downloaded from the above website. All the examples are tested on a PC with 3.0GHz Pentium CPU + 1GB RAM. Note that since we generate the results of ABF by *Graphite*, the computational time on ABF is approximately counted. We also test IP, LSCM, and LPFB+IP by a sparse direct solver – SuperLU [6] on a refined Squirrel model with 435K faces and 220K vertices (see the last row in Table 1). Our method also performs better than IP and LSCM on this solver. In summary, both the effectiveness and the efficiency requirements have been satisfied by our approach.

The segmentation algorithm based on LPFB has been tested on several models, where the small patches generated in the error controlled VSA step can be successfully merged into larger nearly developable patches. The first test is given on the moai model (see Fig.2), where the result from error-controlled VSA consists of 68 small patches. With the help of LPFB, our algorithm finally merges them into 7 nearly developable patches that can be flattened without overlap. The second test (see Fig.10) is on a CSG-like model – every surface on it is planar. Our merging algorithm generates a result with 5 pieces out of 16. The last two tests are given on models with more complex geometry (also shown in Fig.10). Our algorithm merges 78 small patches into 7 on the squirrel model and joins 61 pieces into 11 patterns on the bunny rabbit finally. In summary, the patch merging criteria based on the distortion on the result of LPFB+IP work well with the quasi-developable mesh segmentation algorithm. One interesting application of the atlases generated in our approach is to make toys from them by either paper or textile (see Fig.10).

The two examples shown in Fig. 11 are to compare our segmentation algorithm with the most closely related segmentation approach in [43]. One drawback of [43] is that it is hard to determine the number of charts as the input of their algorithm. Giving small number of charts may yield bad segmentation results. For example, when using the same number of charts in our segmentation results (i.e., 5 for the CSG-like model and 7 for the squirrel), the output of [43] is as shown in Fig. 11(a). Using more patches can reduce the distortion in some degree (see Fig. 11(b)); however, this will greatly increase the difficulty of the stitching job in the applications like toy fabrication.

The authors in [18] presented a method to determine the free boundary for parameterization which is similar to ours, but they did not preserve the length of boundary edges. They

TABLE 1
COMPUTATIONAL STATISTICS

Model	Face No.	Vertex No.	Method	Figure	L^2	$A(\Omega)$	Π_L	Π_θ	Time (s)
Cylinder	6,239	317 / 3,279	IP	Fig.7	1.68	13.5 %	1.14	8.9×10^{-3}	2.41
			LSCM	Fig.7	1.75	49.6 %	0.26	1.3×10^{-2}	1.45
			ABF	Fig.7	1.00	0.28 %	1.3×10^{-3}	5.6×10^{-4}	~ 4
			LPFB+IP	Fig.1	1.00	0.59 %	0.0	1.2×10^{-4}	0.49 (0.001)
Cube	7,768	80 / 425	IP	Fig.7	1.59	30.4 %	3.8×10^{-3}	3.9×10^{-2}	0.062
			LSCM	Fig.7	1.88	44.9 %	0.23	3.9×10^{-2}	0.031
			ABF	Fig.7	1.43	37.9 %	0.21	3.9×10^{-2}	< 1
			LPFB+IP	Fig.1	1.88	50.9 %	0.0	3.9×10^{-2}	0.031 (0.001)
Two-Hole Patch	1,128	198 / 662	IP	Fig. 8	1.02	10.1 %	1.6×10^{-3}	9.4×10^{-3}	0.14
			LSCM	Fig. 8	1.05	10.8 %	5.4×10^{-3}	9.8×10^{-3}	0.094
			ABF	Fig. 8	1.05	9.0 %	4.5×10^{-2}	1.1×10^{-2}	< 1
			LPFB+IP	Fig. 5(e)	1.04	9.5 %	0.0	8.5×10^{-3}	0.047 (0.001)
Squirrel	27,542	1,448 / 14,496	IP	Fig. 9	1.74	15.4 %	0.44	3.9×10^{-2}	68.27
			LSCM	Fig. 9	1.06	7.91 %	4.2×10^{-2}	6.4×10^{-3}	108.27
			ABF	Fig. 9	1.01	2.41 %	1.2×10^{-2}	5.9×10^{-3}	~ 24
			LPFB+IP	Fig. 9	1.01	0.10 %	0.0	1.6×10^{-3}	6.78 (0.016)
Refined Squirrel (with the solver – SuperLU [6])	435,066	5,762 / 220,415	IP	-	1.25	8.74%	0.31	3.2×10^{-3}	43.76
			LSCM	-	1.17	3.74%	0.25	3.4×10^{-3}	45.39
			LPFB+IP	-	1.02	0.05%	0.0	8.4×10^{-4}	22.23 (0.078)

* In the column of vertex number, the previous value is the number of boundary vertices while the later one is the number of all vertices. IP means the intrinsic parameterization with free boundary [7], LSCM is for the least squares conformal maps [19], ABF represents the angle-based flattening [34], and LPFB + IP denotes the computation of length-preserved free boundary followed by the intrinsic parameterization with fixed boundary (i.e., our approach). The time listed in brackets is the time for computing LPFB only.

conducted a 2D polygon morphing technique [31] to let the free boundary mimic the boundary on 3D mesh surface. We find that their method does not work well on the 3D meshes whose boundaries are far from planar (e.g., the examples in Fig. 12). Besides, their approach does not prevent the local self-intersection, which yields great distortions as shown in the left model in Fig. 12. Our LPFB does not have these drawbacks.

5.1 Limitations

Serving as a model to compute the distortion criterion for developable mesh segmentation, the resultant mesh from LPFB+IP needs to be robustly computed. The computation method presented in this paper relies on the surface inner angles on boundary vertices since they are the reference angles to reach in the numerical optimization framework. If the boundary of a segmented patch contains vertices with high curvature, the result of LPFB may be weird. However, this has been naturally avoided by VSA – the first step of our segmentation algorithm. Besides, in practice, there is no guarantee that noises will not be shown on the boundary of a surface patch. Therefore, the first limitation of our approach is that the method for computing LPFB is not robust enough. This is also the reason why the boundary denoising (step 3) is needed in our segmentation algorithm. Tests show that our method works well if the boundary denoising step is applied together.

The second limitation of our approach is that: although the constraints derived from the closed-path theorem has been added to the numerical optimization framework so that the local-self-intersection (e.g., the case shown in Fig.13(a)) is prevented, there is no constraint for avoiding the global-self-intersection on the computed LPFB (e.g., Fig.13(b)). Therefore, in the measurements for classification, we need the $S(\Omega)$ term. The method for preventing global-intersection is still under investigation. The angle expansion method for self-intersection from [34] seems to be a good candidate.

Thirdly, finding the thresholds for the classifiers in our segmentation algorithm is by no means an easy job. We determine them by supervised learning: samples of nearly developable surface and non-developable surface are first selected to train the classifier. The values of $L^2(\Omega)$ and $A(\Omega)$ are computed on LPFB+IP of the sample patches. Then, the thresholds are drawn so that most samples can be classified into a correct category. Therefore, the thresholds depend on the samples employed to train the classifier which is not robust enough.

Lastly, the toy fabrication and the texture mapping applications may wish to have patterns with smooth boundaries. Therefore, we plan to replace the 2nd step of our segmentation algorithm by a new method that can generate more smooth boundaries.

6 CONCLUSION

This paper presents the method about how to efficiently compute a length-preserved free boundary (LPFB) for a given mesh surface patch, and employs it to speed up the following intrinsic parameterization (IP). Therefore, distortions generated on the flattening result are used as criteria to detect whether a given patch is quasi-developable. The computation of LPFB is formulated as a numerical optimization problem in the angle space, where we are trying to optimize the angle excesses on the boundary while preserving the constraints derived from the closed-path theorem

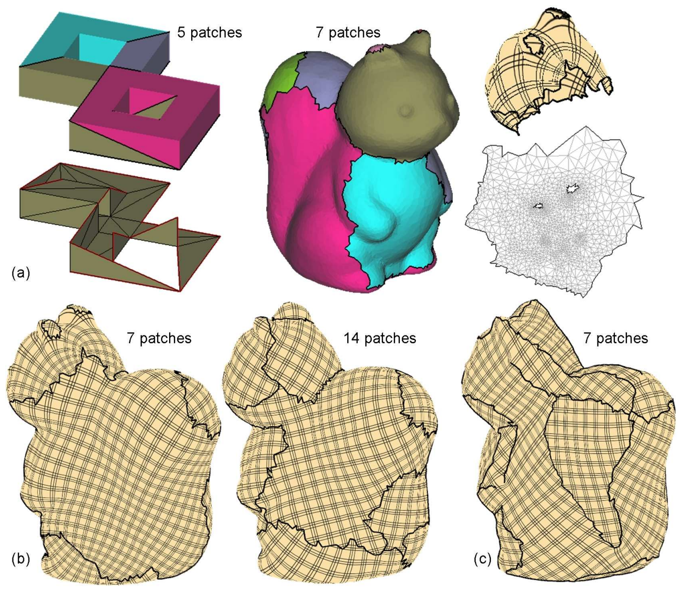


Fig. 11. It is difficult to determine the number of charts by [43]. (a) The results by using 5 pacthes for the CSG-like model and 7 patches for the squirrel model, which are the same as our segmentation results in Fig. 10 – it is easy to find that the resultant segmentation on the CSG-like model is non-flattenable as there is topological obstruction (left) and great distortions are shown at the head of squirrel (right). (b) Distortion on the squirrel model can be reduced by increasing the number of patches from 7 to 14, but still shows large distortion at the patch near foot. (c) Our segmentation result can have less distortion on 7 patches.

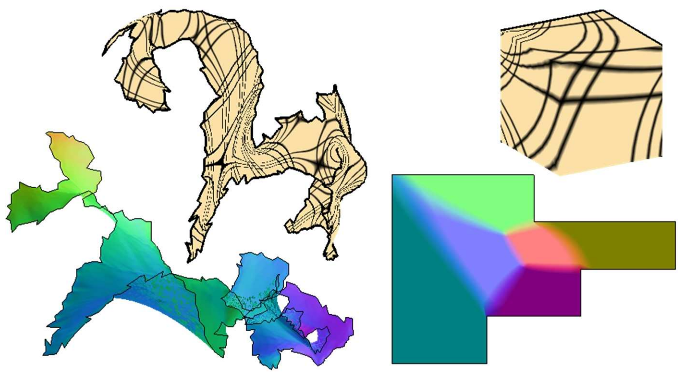


Fig. 12. The flattening results with the free boundaries that are determined by the 2D polygon morphing method in [18].

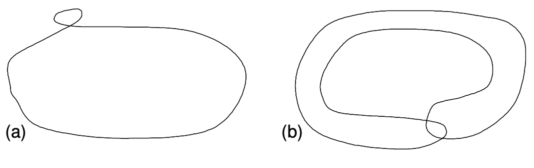


Fig. 13. Two cases for self-intersection: (a) local-self-intersection which has been prevented by the constraints from the closed-path theorem, and (b) global-self-intersection which has to be detected after computing the length-preserved free boundary.

and the length preservation. By introducing a virtual-cutting scheme, the method has been further extended to overcome the limitation of disk-like topology. Based on LPFB+IP, a trial-and-error mesh segmentation has been developed in this paper to partition a given model into nearly developable atlases. Numerous examples have been tested on the computation method for LPFB and the segmentation algorithm to demonstrate the effectiveness and the efficiency of our approach.

Acknowledgment

The authors would like to thank the AIM@SHAPE Shape Repository for sharing some of the models in this paper.

Appendix

B_λ, B_θ and Λ in Eq. (12) can be efficiently evaluated.

Statement 1 B_λ is computed by

$$B_\lambda = \left(-\frac{\partial J}{\partial \lambda_\theta}, -\frac{\partial J}{\partial \lambda_{0x}}, -\frac{\partial J}{\partial \lambda_{0y}}, \cdots, -\frac{\partial J}{\partial \lambda_{mx}}, -\frac{\partial J}{\partial \lambda_{my}}\right)$$

where

$$-\frac{\partial J}{\partial \lambda_\theta} = -(n-2)\pi + \sum_{k=1}^{n} \theta_k ,$$

$$-\frac{\partial J}{\partial \lambda_{px}} = -\sum_{k=\alpha(p)}^{\beta(p)-1} l_k \cos\phi_k \text{ , and } -\frac{\partial J}{\partial \lambda_{py}} = -\sum_{k=\alpha(p)}^{\beta(p)-1} l_k \sin\phi_k .$$

Statement 2 $B_\theta = \{-\partial J / \partial \theta_i\}$ can be efficient evaluated by the following recursion formulas.

$$b_{\theta_1} = -(\theta_1 - a_1) + \lambda_\theta + \sum_{p=0}^{m} \sum_{k=\alpha(p)}^{\beta(p)-1} (-\lambda_{px} \sin\phi_k + \lambda_{py} \cos\phi_k) l_k ,$$

$$b_{\theta_{i+1}} = b_{\theta_i} + (\theta_i - a_i) - (\theta_{i+1} - a_{i+1}) + \sum_{p=0}^{m} A(p,i) ,$$

with

$$A(p,i) = \begin{cases} (\lambda_{px} \sin\phi_i - \lambda_{py} \cos\phi_i) l_i, & \alpha(p) \le i < \beta(p) \\ 0, & otherwise \end{cases} .$$

Statement 3 For Λ, whose dimension is $(2m+3) \times n$, its i-th column vector is

$$\Lambda_i = \frac{\partial^2 J}{\partial \lambda \partial \theta_i} = \begin{pmatrix} \frac{\partial}{\partial \theta_i}((n-2)\pi - \sum_{k=1}^{n} \theta_k) \\ \frac{\partial}{\partial \theta_i} \sum_{k=\alpha(p)}^{\beta(p)-1} l_k \cos\phi_k \\ \frac{\partial}{\partial \theta_i} \sum_{k=\alpha(p)}^{\beta(p)-1} l_k \sin\phi_k \\ \cdots \end{pmatrix}_{(2m+3)\times 1} ,$$

with $p = 0,1,\cdots,m$. Every element of Λ_i can be evaluated by

$$\Lambda_{1,i} = -1 ,$$

$$\Lambda_{2p+2,i+1} = \Lambda_{2p+2,i} + B(p,i) ,$$

$$\Lambda_{2p+3,i+1} = \Lambda_{2p+3,i} + D(p,i) ,$$

with

$$B(p,i) = \begin{cases} -l_i \sin\phi_i, & a(p) \le i < \beta(p) \\ 0, & otherwise \end{cases} ,$$

$$D(p,i) = \begin{cases} l_i \cos\phi_i, & a(p) \le i < \beta(p) \\ 0, & otherwise \end{cases} .$$

REFERENCES

[1] P.N. Azariadis and N.A. Aspragathos, "Design of Plane Development of Doubly Curved Surface," *Computer-Aided Design*, vol. 29, pp. 675–685, 1997.

[2] M. Aono, D.E. Breen and M.J. Wozny, "Modeling Methods for the Design of 3D Broadcloth Composite Parts," *Computer-Aided Design*, vol. 33, pp. 989–1007, 2001.

[3] M. Aono, D.E. Breen and M.J. Wozny, "Fitting a Woven-Cloth Model to a Curved Surface: Mapping Algorithms," *Computer-Aided Design*, vol. 26, pp. 278–292, 1994.

[4] D. Cohen-Steiner, P. Alliez, and M. Desbrun, "Variational Shape Approximation," *ACM Trans. Graphics*, *SIGGRAPH 2004*, vol.23, pp. 905-914, 2004.

[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms* (2nd ed.), MIT Press, 2001.

[6] J.W. Demmel, S.C. Eisenstat, J.R. Gilbert, X.S. Li, and J.W.H. Liu, "A supernodal approach to sparse partial pivoting," *SIAM Journal on Matrix Analysis and Applications*, vol.20, no.3, pp.720–755, 1999.

[7] M. Desbrun, M. Meyer, and P. Alliez, "Intrinsic Parameterizations of Surface Meshes," *Computer Graphics Forum*, *Proc. of Eurographics 2002*, vol. 21, pp. 209–218, 2002.

[8] M.P. Do Carmo, *Differential Geometry of Curves and Surfaces*, Englewood Cliffs, NJ, Prentice Hall, 1976.

[9] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification* (2nd Edition), pp.20-160, Wiley: New York, 2001.

[10] M.S. Floater and K.Hormann, "Surface Parameterization: a Tutorial and Survey," *Advances in Multiresolution for Geometric Modelling*, N.A. Dodgson, M.S. Floater, and M.A. Sabin (eds.), Springer-Verlag, Heidelberg, pp.157-186, 2005.

[11] X. Gu, S.-T. Yau, "Global Conformal Surface Parameterization," *Proc. of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pp.127-137, 2003.

[12] K. Hormann, and G. Greiner, "MIPS: An Efficient Global Parametrization Method," *Curve and Surface Design Saint-Malo 1999* (P.-J. Laurent, P. Sablonnire, and L. Schumaker eds.), pp.153-162, Vanderbilt University Press.

[13] D. Julius, V. Kraevoy, and A. Sheffer, "D-Charts: Quasi-Developable Mesh Segmentation," *Computer Graphics Forum*, *Proc. of Eurographics 2005*, vol. 24, pp. 581-590, 2005.

[14] Z. Karni, C. Gotsman, and S.J. Gortler, "Free-Boundary Linear Parameterization of 3D Meshes in the Presence of Constraints," *Proc. of Shape Modeling International*, pp. 266-275, June, 2005.

[15] S. Katz, G. Leifman, and A. Tal, "Mesh Segmentation using Feature Point and Core Extraction," *The Visual Computer*, *Special Issue of Pacific Graphics 05*, vol. 21, no. 8-10, pp. 649-658, 2005.

[16] S. Katz and A. Tal, "Hierarchical Mesh Decomposition using Fuzzy Clustering and Cuts," *ACM Trans. Graphics*, *Proc. SIGGRAPH 2003*, vol.22, no. 3, pp. 954-961, 2003.

[17] V. Kraevoy, A. Sheffer, and C. Gotsman, "Matchmaker: Constructing Constrained Texture Maps," *ACM Trans. Graphics*, *Proc. SIGGRAPH 2003*, vol.22, no. 3, pp. 326-333, 2003.

[18] Y. Lee, H-S Kim, and S. Lee, "Mesh Parameterization with a Virtual Boundary," *Computers & Graphics*, vol. 26, pp. 677-686, 2002.

[19] B. Levy, S. Petitjean, N. Ray and J. Maillot, "Least Squares Conformal Maps for Automatic Texture Atlas Generation," *ACM Trans. Graph*, *SIGGRAPH 2002*, vol. 21, pp. 362–371, 2002.

[20] J. Maillot, H. Yahia, and A. Verroust, "Interactive Texture Mapping," *Proc. SIGGRAPH 93*, pp. 27-34, 1993.

[21] J. Mitani and H. Suzuki, "Making Papercraft Toy from Meshes Using Strip-based Approximate Unfolding," *ACM Trans. Graphics*, *SIGGRAPH 2004*, vol.23, no.3, pp259-263, 2004.

[22] J. McCartney, B.K. Hinds and B.L. Seow, "The Flattening of Triangulated Surfaces Incorporating Darts and Gussets," *Computer-Aided Design*, vol. 31, pp. 249–260, 1999.

[23] M. Meyer, M. Desbrun, P. Schroder, and A.H. Barr, "Discrete Differential-Geometry Operators for Triangulated 2-Manifolds," *Proc. of Visualization and Mathematics*, 2002.

[24] M.E. Mortenson, *Geometric Modeling* (2nd Edition), pp.282-310, Wiley: New York, 1997.

[25] J. Nocedal, and S.J. Wright, *Numerical Optimization*, Springer-Verlag, 1999.

[26] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge: Cambridge University Press, pp.71-89, 1995.

[27] P. Sander, S. Gortler, J. Snyder, and H. Hoppe, "Signal-Specialized Parametrization," *Eurographics Workshop on Rendering 2002*, pp.87-100, 2002.

[28] P.V. Sander, J. Snyder, S.J. Gortler, and H. Hoppe, "Texture Mapping Progressive Meshes," *Proc. of SIGGRAPH 2001*, pp. 409-416, 2001.

[29] P.V. Sander, Z. Wood, S.J. Gortler, J. Snyder, H. Hoppe, "Multi-Chart Geometry Images," *Proc. of First Symposium on Geometry Processing 2003*, pp. 146-155, 2003.

[30] L. Saroul, O. Figueiredo, and R.D. Hersch, "Distance Preserving Flattening of Surface Sections," *IEEE Trans. Visualization and Computer Graphics*, vol.12, pp.26-35, 2006.

[31] T.W. Sederberg, P. Gao, G. Wang, H. Mu, "2-D shape blending: an intrinsic solution to the vertex path problem," *Proc. of SIGGRAPH'93*, pp.15-18, 1993.

[32] I. Shatz, A. Tal, and G. Leifman, "Paper Craft Models from Meshes," *The Visual Computer*, *Pacific Graphics 2006*, vol.22, no.9-11, pp.825-834, 2006.

[33] A. Sheffer, "Spanning Tree Seams for Reducing Parameterization Distortion of Triangulated Surfaces," *Proc. of International Conference on Shape Modeling and Applications 2002*, pp. 61-66, 2002.

[34] A. Sheffer, and E. de Sturler, "Parameterization of Faceted Surfaces for Meshing Using Angle Based Flattening," *Engineering with Computers*, vol. 17, no. 3, pp. 326-337, 2001.

[35] A. Sheffer, B. Lévy, M. Mogilnitsky, and A. Bogomjakov, "ABF++: Fast and Robust Angle Based Flattening," *ACM Tran. Graphics*, vol. 24, no. 2, pp. 311-330, 2005.

[36] G. Stylianou and G. Farin, "Crest Lines for Surface Segmentation and Flattening," *IEEE Trans. Visualization and Computer Graphics*, vol.10, pp.536-544, 2004.

[37] O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski, "Bounded-distortion Piecewise Mesh Parameterization," *Proc. of IEEE Visualization 2002*, pp.355-362, 2002.

[38] G. Taubin, "Linear Anisotropic Mesh Filtering," *Technical Report of IBM Research*, TR-RC2213, 2001.

[39] C.C.L. Wang, S.S.F. Smith, and M.M.F. Yuen, "Surface Flattening Based on Energy Model," *Computer-Aided Design*, vol.34, no.11, pp.823-833, 2002.

[40] C.C.L. Wang, and K. Tang, "Achieving Developability of a Polygonal Surface by Minimum Deformation: a Study of Global and Local Optimization Approaches," *The Visual Computer*, vol.20, pp.521-539, 2004.

[41] C.C.L. Wang, K. Tang, and B.M.L. Yeung, "Freeform Surface Flattening by Fitting a Woven Mesh Model," *Computer-Aided Design*, vol. 37, pp. 799-814, 2005.

[42] H. Yamauchi, S. Lee, Y. Lee, Y. Ohtake, A. Belyaev, and H.-P.

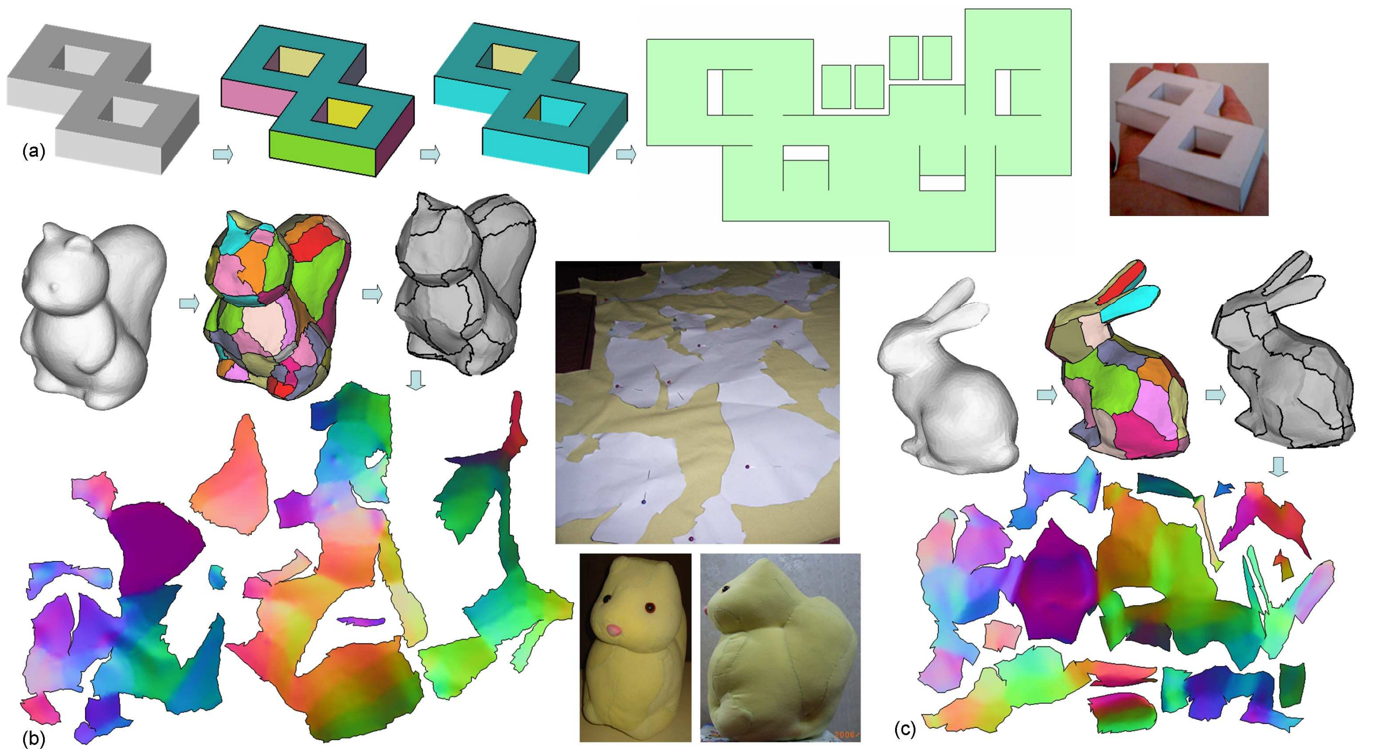


Fig. 10. Testing our mesh segmentation algorithm on three models: (a) a CSG-like model, (b) a squirrel model, and (c) a bunny rabbit. The results are given in three steps. The colorful segmentations of given models are the results from error-controlled VSA, and then the results after LPFB-based patch merging are given in gray. The flattening layout is shown in colors which represent normal vectors on the original 3D models. Finally, the photographs for the physical models make by these patterns are shown.

Seidel, "Feature Sensitive Mesh Segmentation with Mean Shift," *Proc. Shape Modeling International 2005*, pp. 236-243, 2005.

[43] H. Yamauchi, S. Gumhold, R. Zayer, and H.-P. Seidel, "Mesh Segmentation Driven by Gaussian Curvature," *The Visual Computer*, vol. 21, no.8-10, pp. 659-668, 2005.

[44] J Yan, X. Yang, P. Shi, and D. Zhang, "Mesh Parameterization by Minimizing the Synthesized Distortion Metric with the Coefficient-Optimizing Algorithm," *IEEE Trans. Visualization and Computer Graphics*, vol.12, pp. 83-92, 2006.

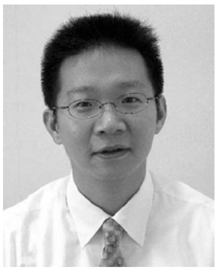
[45] S. Yoshizawa, A.G. Belyaev, and H.-P. Seidel, "A Fast and Simple Stretch-Minimizing Mesh Parameterization," *Proc. Shape Modeling and Applications*, pp. 200-208, June 7-11, 2004, Genova, Italy.

[46] R. Zayer, C. Rössl, and H.-P. Seidel, "Setting the Boundary Free: A Composite Approach to Surface Parameterization," *Proc. of the Third Eurographics Symposium on Geometry Processing*, pp.91-100, 2005.

[47] E. Zhang, K. Mischaikow, and G. Turk, "Feature-based Surface Parameterization and Texture Mapping," *ACM Trans. Graphics*, vol. 24, no. 1, pp. 1-27, 2005.

[48] K. Zhou, J. Synder, B. Guo, H.-Y. Shum, "Iso-charts: Stretch-driven Mesh Parameterization using Spectral Analysis," *Proc. Eurographics Symposium on Geometry Processing*, Nice, France, July, 2004.

[49] G. Zigelman, R. Kimmel, and N. Kiryati, "Texture Mapping Using Surface Flattening via Multidimensional Scaling," *IEEE Trans. Visualization and Computer Graphics*, vol.8, pp.198-207, 2002.



Charlie C. L. Wang is currently an Assistant Professor at the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong. He gained his B.Eng. (1998) in Mechatronics Engineering from Huazhong University of Science and Technology, M.Phil. (2000) and Ph.D. (2002) in Mechanical Engineering from The Hong Kong University of Science and Technology. He is a member of IEEE and ASME. His current research interests include geometric modeling in computer-aided design and manufacturing, biomedical engineering, and computer graphics, as well as computational physics in virtual reality.